VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vizualizace dynamických sliderů v prostředí webu

Visualization of Dynamic Sliders on Web

2016 Tomáš Podolák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Podolák**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vizualizace dynamických sliderů v prostředí webu
Visualization of Dynamic Sliders on Web

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je navrhnout a implementovat komponentu pro dynamický slider pro webové stránky s využitím moderních technologií v oblasti vývoje webových prezentací, jako HTML5, CSS3 a jQuery. Z pohledu frontendové části aplikace se bude jednat o responzivní prezentaci. Použitou technologií bude Microsoft .NET.

Body zadání:
1. Přehled existujících řešení v oblasti nabízených API pro prezentaci dynamických sliderů a jejich tvorbu.
2. Návrh vhodné datové struktury a modulu pro prezentaci sliderů s využitím technologie MS.NET.
3. Implementace a frontendové části pro prezentace a administrační aplikace pro návrh sliderů.
4. Provedení experimentů a stanovení limitů navržené aplikace.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] P. Lubbers, HTML5 - Programujeme moderní webové aplikace, 2011, CPRESS, ISBN: 9788025135396
[2] Brian P. Hogan, HTML5 a CSS3 - Výukový kurz webového vývojáře, 2011, CPRESS, ISBN: 9788025135761
[3] P.Gasston, The Book of CSS3: A Developer's Guide to the Future of Web Design, 2011, No STarch Press, ISBN: 978-1593272869

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Gajdoš, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2016 .....

Ďakujem vedúcemu mojej bakalárskej práce Ing. Petru Gajdošovi, Ph.D., za jeho odborné vedenie, podnety, rady, pripomienky a pomoc pri jej spracovaní.

Abstrakt

Cieľom záverečnej práce bolo vizuálne zobrazenie konceptu na webovej ploche. Objektom nášho skúmania boli elementy, ktoré prezentujú text, obrázok a video. Tieto prvky sú nezávislé od seba, ale závislé na svojej ploche, kde sa nachádzajú. Vypracovali sme návrh na zachytenie vlastností prvkov v určitý čas a výsledkom práce je vytvorenie aplikácie na pridávanie a následnú editáciu týchto prvkov. Táto bakalárska práca vizuálne prezentuje výsledok editácie ako dynamického celku. Je rozdelená do ôsmich kapitol a jednej prílohy. Aplikácia je pomenovaná Dynamic Slide Show, čo v preklade znamená „dynamická posuvná prezentácia". Ide o sériu stránok s dôležitými informáciami zobrazenými na displeji. Každá snímka má svoju dobu zobrazenia a dynamické vlastnosti, ktoré sa postupne zobrazujú. Výsledok môže slúžiť na prezentovanie užívateľovej témy, ale aj na vytvorenie reklamy na internete. Dynamické vizuálne zobrazenie púta najmä pozornosť užívateľa, preto je pravdepodobné, že sa aplikácia Dynamic Slide Show bude využívať aj po prezentovaní a odovzdaní. V mojom prípade vidím využitie aplikácie na vizualizáciu reklám na webových stránkach, na prezentovanie produktu, témy a podobne.

Klíčová slova: Dynamická vizualizácia, Posuvná prezentácia, asp.net applikácia

Abstract

The main aim of the bachelor thesis is visual representation of a concept on the web site. The objects of our research are elements, which represent the text, picture and video. These elements are independent from each other, but on the other hand, they depend on each other on the web site. We worked out a suggestion how to find the features of elements at the certain time point and the result is the creation of an application for adding and editing of these elements. Thesis is the visual representation of the result of editing as dynamical activity. Thesis is divided into eight chapters and one attachement. Application is called Dynamic Slide Show. It includes series of web sites with important information, which is on display. Each picture is represented for certain time period and has dynamic features, which show up gradually. The result can serve for presenting the user's topic or for creating an advertisement on the internet. Dynamic visual presentation attracts user's attention, so application Dynamic Slide Show will be probably used

after the presenting. In this case, the application can be used for advertisement visualisation on the web pages, for presenting the product, topic and etc.

Key Words: Dynamic visualization, Slide Show, asp.net application

Obsah

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript + XML
API	–	Application Programming Interface
ASP	–	Active Server Pages
CSS	–	Cascading Style Sheets
HTML	–	Hyper Text Markup Language
JS	–	JavaScript
JSON	–	JavaScript Object Notation
XML	–	Extensible Markup Language

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

S postupne rastúcim vývojom webových aplikácií vzniká potreba prezentovať nové veci tak, aby to malo prínos do budúcnosti. Vo všeobecnosti platí, že tri mesiace na Internete sú zhruba rok ľudského života, tým je charakterizovaný i nezastaviteľný vývoj. To je dôvod, prečo weboví vývojári neustále objavujú nové techniky pre svoju prácu. Ešte prednedávnom to vyzeralo tak, že HTML5 a CSS3 sú témou vzdialenej budúcnosti, no vďaka ich rýchlemu vývoju používajú tieto technológie vývojárske spoločnosti. HTML5 a CSS3 navyše podporujú prehliadače ako Opera, FireFox, Safari, Google Chrome.[1]

Bakalársku prácu s názvom Vizualizácia dynamických sliderov na webe som si vybral preto, lebo jej fungovanie je prínosom. Nech sa v modernom svete nachádzame kdekoľvek, ako prvé našu pozornosť zaujme vizuálne zobrazenie dát. Takýmito dátami môžeme chápať zobrazenie reklamy na bilbordoch, grafické zobrazenie štatistík, prehrávanie hudby a mnoho ďalších. Keď sa k takejto prezentácii pridajú dynamické efekty, je zjavné, že pozornosť sa o to viac navýši. Typické príklady sú viditeľné i napr. v moderných obchodných centrách, kde je takáto reklama skoro všade. Ak sa zameriame na webové stránky, môže byť takáto pozornosť získaná obdobným spôsobom. Postup je stále rovnaký, či už ide o prezentovanie produktu, udalosti, umenia alebo niečoho podobného.

Počas prvej konzultácie mi vedúci bakalárskej práce prezentoval možnosti, ciele a využiteľnosť práce do budúcna, čo ma len utvrdilo v tom, že táto práca bude mať široké využitie. Sú v nej riešené možnosti vývoja moderných vizuálnych prezentácií, no zaoberá sa i problémami, ktoré sa môžu vyskytnúť pri ich tvorbe. Hlavný cieľ je vyriešiť proces zobrazovania, prezentácie, pridať elementy a pracovať s nimi tak, aby podporovali dynamické vlastnosti vizualizácie – rotovanie textu, zmena farby a fontu písma, dynamickosť obrázkov. Druhá kapitola sa venuje využitiu konkurenčných softvérov, použiteľnosti knižníc pre tvorbu vizualizácie a možnostiam ich využitia. V tretej kapitole je uvedený dôvod výberu konkrétnej technológie pre tvorbu našej aplikácie, no je tu i stručne načrtnutá práca s danou technológiou. Vo štvrtej kapitole je predstavená naša aplikácia. Piata kapitola rieši úpravy vizualizácie užívateľom. Šiesta kapitola sa zameriava na konečné zobrazenie dynamickej vizualizácie. Siedma kapitola vyhodnocuje testy danej aplikácie naprieč prehliadačmi.

2 Konkurencia a techniky vizualizácie

V tejto kapitole sa zameriavame na prieskum možností pre vytváranie dynamického obsahu na webe a s ním spojenými výhodami. Zaoberáme sa aj s ich nedostatkami a možnými problémami. Prieskumom myslíme vyhľadávanie podobných aplikácií a knižníc na tvorbu vizuálneho a dynamického konceptu. Na webových stránkach sa môžu zobrazovať rôzne druhy dát – mapy, obrázky, grafy, texty, vedomosti v odbore umenia, hudby, vzdelania a podobne. Na každý druh obsahu sa špecializujú rozdielne softvéry alebo knižnice slúžiace pre webových dizajnéroch.

2.1 Konkurenčný softvér

Microsoft Power Point

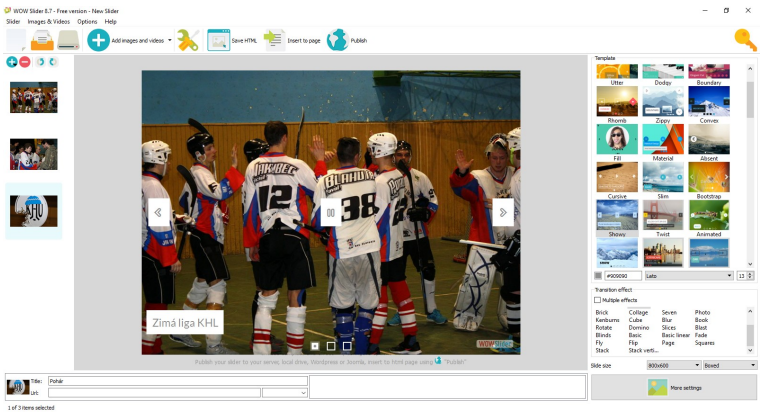
Ide o produkt od spoločnosti Microsoft ako súčasť balíčka Microsoft Office pre Microsoft Windows a Mac OS užívateľov. Môžeme konštatovať, že je to najpoužívanejší program pre vytváranie posuvnej prezentácie špecifického obsahu. Spoločnosť umožňuje študentom používať Office balíček zadarmo. Komerční a domáci užívatelia si za tieto služby musia zaplatiť. Tento produkt podporuje hyperlinky, úpravu, animáciu textu a obrázkov.[2] Čo sa týka použiteľnosti, z hľadiska webu nejde o užitočný program. Pre prezentáciu na webe je nutné využiť konvektory, ale výsledok bude obsahovať iba obrázok, ktorý sa bude zobrazovať bez vnútorných efektov. Pre dynamickosť sa môže použiť CSS, ale prejavovať sa bude na celku, a nie vo vnútorných prvkoch. Vizualizovať môžeme hlavne text a obrázky. Hudba môže byť len prehrávaná na pozadí.

Obrázek 1: Microsoft Power Point 2013

Wow Slider

WOW Silder sa reprezentuje ako editor na rýchle vytváranie Slide Show. Podstatou programu je výber obrázkov a ich pridanie do zoznamu. Ku každému obrázku je pridaný popis a

z preddefinovaných efektov prechodu si užívateľ jednoducho vyberie podľa seba. Program má v sebe funkciu, po ktorej sa na vybrané miesto uloží vygenerovaný kód, ktorý pozostáva zo stajlovania v CSS, import javascriptov a samotného HTML kódu. Stiahnuť sa dá pre platformy MS Windows a Mac. Po stiahnutí a nainštalovaní je po určitú dobu program v plnej verzii, avšak po uplynutí skúšobnej doby je potrebné si za ďalšie služby zaplatiť. Pre správne fungovanie využíva svoje vlastné vytvorené scripty.



Obrázek 2: WOW Slider 8.7

Slider Revolution

Slider Revolution je jeden z najznámejších inovatívnych WordPress Pluginov pre tvorbu posuvného obsahu. Dokáže spracovať všetky potrebné prvky, a to text, obrázky aj video. Pre svoje fungovanie využíva jQuery. Rieši už väčšie problémy, ako sú responzivita, udalosti s myškou a pod. Aktuálne najnovšia dostupná verzia je 5.2. Nevýhodou pre bežných užívateľov je, že za každý projekt sa musí platiť osobitne. [3]

Wolfram Alpha

Ide o internetovú službu, ktorá je kombináciou výpočtov, vyhľadávania a encyklopédie v jednom celku. Zameriava sa hlavne na vedecký a matematický odbor, ale obsahuje aj vedomosti a dáta v odbore histórie, geografie, umenia, hudby a i. V roku 2009 si vyslúžila táto internetová služba za svoje výsledky a užitočnosť ocenenie inovácie roka, čím si získala prezývku „knowledge engine“ – v preklade vedomostný stroj. Hlavnou úlohou je spracovanie problému, vytvorenie riešenia a jeho vizuálne zobrazenia vo forme grafov, rovnice, algoritmov a štatistík. Všetky vedomosti si uchováva vo svojom zdroji, ktorý je reprezentovaný vlastnou databázou. Práca užívateľa so službou je veľmi jednoduchá. Postačí si zadať v prehliadači oficiálnu stránku http://www.wolframalpha.com a vo vstupnom boxe položiť otázku v ľudskom jazyku alebo ma-

tematickou formou – pomocou rovnice, funkcie a podobne.[4]

Obrázek 3: Wolfram Alpha

2.2 Dostupné knižnice

Postupným vývojom webových aplikácií vznikla potreba vytvárať samostatne posuvné prezentácie a s nimi spojené dynamické prvky. Vďaka pokračujúcemu vývoju a zdokonaľovaniu máme dnes na výber viaceré knižnice, ktoré zhrnieme v tejto kapitole.

Impress.js

Ide o prezentačný framework založený na využívaní transformácií pomocou stajlovania v CSS. Je to moderná JavaScriptová knižnica, ktorá komunikuje s HTML a CSS. Zvláda zložité prechody medzi snímkami.[5] Na internete je dostupných viacero kurzov pre tento framework, ktorý je voľne dostupný. Práca s Impress.js je jednoduchá, pričom je prvoradé importovanie javascriptovej knižnice a štýlov CSS, vyčlenenie kontextu na prezentáciu pomocou DIV. Následne sa zadá ručne HTML kódom počet slidov s vlastnosťami, ktoré sú potrebné na prechody slidov. Parametrami môžu byť x, y, rotácia a rôzne efekty.

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link href="styles.css" rel="stylesheet" />
```

```
</head>

<body>
<div id="impress">
<div class="step" data-x="0" data-y="0">
This is my first slide. I start at 0,0. Welcome.
</div>
<div class="step" data-x="500" data-y="-800" data-scale="0.5">
This is slide 2. It slides left 500 pixels.
</div>
<div class="step" data-x="-2600" data-y="-800" data-rotate-x="30" data-rotate-y
    ="-60" data-rotate-z="90" data-scale="4">
This is slide 3. It slides down 400 pixels.
</div>
</div>
<script type="text/javascript" src="impress.js"></script>
<script>impress().init();</script>
</body>
```

Výpis 1: Ukážka práce s Impress.js

Reveal.js

Ide o HTML prezentačný framework. Slúži na tvorbu interaktívnej plochy za použitia HTML. Jednotlivé snímky môžu byť vnorené do vnútra snímky. Prechody sú realizované horizontálnym aj vertikálnym pohybom, čím sa odlišuje od ostatných prezentačných frameworkov. Ovládanie prezentácie je možné aj pomocou klávesnice. Pre ľahšiu prácu je ponúkaný na oficiálnej stránke, ktorá ešte stále nie je dokončená, editor na tvorbu prezentácií. Po vytvorení sú plne responzivné. Výhodou je, že využívanie tejto knižnice je zadarmo a nie je potrebné platiť licenciu. Práca s Reveal.js nie je náročná a po prezretí zopár tutoriálov je užívateľ sám schopný tvoriť prezentácie.[6]

D3.js

Je to JavaScriptová knižnica pre tvorbu interaktívnej a dynamickej vizualizácie dát vo webových prehliadačoch. Obsahuje v sebe vopred predvypracované JavaScriptové funkcie pre vytváranie a ovládanie prvkov, vytvorenie SVG objektov, definovanie štýlov, dynamické efekty a podobne. Dáta môže obsahovať vo viacerých formátoch, najčastejšie ide o JSON, hodnoty oddelené čiarkou (CSV). Slúži na manipuláciu a vizualizáciu dát, preto je označovaná ako Data-

Driven Document. Jej najväčšou výhodou je to, že dokáže využívať SVG, HTML5 a CSS dokopy. SVG je vektorovo orientovaná grafika v podobnom formáte ako XML. Od ostatných javascriptových knižníc sa líši tým, že má kontrolu nad finálnym výsledkom. Začiatok svojho vývoja datuje okolo roku 2011.[7] Túto knižnicu sme si vybrali pre vývoj našej webovej aplikácie páve pre spomínané výhody, nakoľko umožňuje i všetky potrebné manipulácie s dátami, ktoré budeme využívať. Základnej práci s D3 sa budeme venovať v nasledujúcej podkapitole.

Chart.js

Je to moderná JavaScriptová knižnica na spracovanie a vizuálne zobrazovanie rôznych typov grafov prostredníctvom HTML5, a to pomocou elementu canvas. V súčasnosti je ešte vo vývoji. Podporuje šesť typov grafov: čiarový, stĺpcový, radarový, kruhový, koláčový, polárny. Grafy majú v sebe zabudovanú responzivitu.[8]

Google Charts

API od spoločnosti Google. Funkcionalita tejto služby je vykresľovanie grafov so spojením vlastností HTML a SVG rozhrania. Keďže sa špecializuje na grafy, poskytuje viacero typov grafov – Venový diagram, Korelačný diagram, mapy, stĺpcový, kruhový a mnohé ďalšie. Podporuje aj interakciu grafov so zobrazením detailov určitého úseku grafu. Obsahuje svoju vlastnú dokumentáciu na vytváranie grafov.[8]

Modest Maps.js

Nástroj reprezentujúci API na spracovanie a zobrazovanie máp na webovej stránke. Dokáže integráciu a interaktivitu máp. Základná knižnica môže byť doplnená o prídavné pluginym, čím sa zlepší konečná vizualizácia mapových elementov. Ide o užitočný nástroj pre tvorbu geografických máp.[8]

Web Audio API

Umožňuje ovládať a interpretovať zvuk v prehliadači. Ide o „high-level“ JavaScriptovú API na spracovanie a vizuálne zobrazenie audia na webe. Audio sa spracováva do zvukového grafu, pomocou ktorého sa dokáže vykresľovať. Zachytáva viacero druhov dát: frekvenciu, informácie o čase, priebehu, vytvorenie osciloskopu, hlasový modulátor a mnohé ďalšie. Po uložení a načítaní môžeme začať vizualizovať zvuk.[8] Práca s API je už náročnejšia a užívateľ sa musí riadiť podľa postupov v dokumentácií na W3C.

Flat UI Colors

Ide o internetovú aplikáciu, ktorá ponúka vizualizovanie farby. Jej hlavná funkcionalita je zobrazenie najmodernejších typov farby pre moderné webové aplikácie. Obsahuje v sebe typy farieb, ktoré sa reprezentujú v hexadecimálnom, RGB, RGBA formáte. Stačí kliknúť na farbu a následne ju použiť vo svojej aplikácii. Slúži čisto na vizualizovanie moderných farieb. Pri vývoji bakalárskej práci boli použité farby práve z tejto stránky.

3 D3 a elementy

3.1 Ukážka práce s D3.js

Vytvorenie canvasu, Slection

Po importovaní knižnice je možné jednoducho pridávať obsah na jednotlivé miesta v HTML dokumente pomocou operácie Selection. Môžeme využiť *.select* alebo *.selectAll*, pričom v druhom prípade sa vyberie všetko, čo bude vyhovovať výberu. Využijeme to pri výbere všetkých obrázkov, textu a podobne. Pomocou funkcie *.append* dokážeme pridať SVG plochu, na ktorú budeme môcť pridávať elementy. Vrstva „g“ sa pridáva do canvasu pre špeciálne použitie, ako je napríklad ťahanie elementov po canvase, zoom, ostatné udalosti s myškou a podobne.[9]

```
<script src="http://d3js.org/d3.v3.min.js" charset="utf-8"></script>

<script>
var canvas = d3.select("body")
        .append("svg")
        .attr("width","400px")
        .attr("height","400px")
        .append("g");
</script>
```

Výpis 2: Vytvorenie D3 canvasu

Pridanie elementu text a image do canvasu

Pomocou kľúčových slov „image“ (obrázok) a „text“ môžeme po vyplnení potrebných atribútov pridávať do canvasu jednotlivé elementy. S elementmi môžeme ďalej pracovať aj v ďalších častiach kódu pomocou Selection. Elementom je povolené pridávať vlastné atribúty. Vlastné atribúty budeme využívať na rozdelenie prvkov podľa ich jedinečného identifikačného čísla alebo podľa čísla, do ktorého canvasu patria. Vlastnosti prvkov môžeme behom aplikácie meniť rovnakým spôsobom, ako boli definované. Pre získanie hodnoty atribútu stačí vykonať funkciu .attr s jediným parametrom s názvom vlastnosti.[10]

```
<script>
var text = canvas.append("text")
           .text("Textovy Element")
           .attr("font-family","Verdana")
```

```
            .attr("font-size", 20)
            .style("fill","green")
            .attr("x", "150")
            .attr("y", "300")

var image = canvas.append("image")
            .attr("height", "150px")
            .attr("width", "150px")
            .attr("xlink:href", "myImage.png")
            .attr("x", "350")
            .attr("y", "50")
</script>
```

Výpis 3: Pridávanie elementov do canvasu

3.2 Pridávanie elementu video

Omezení videa v HTML5

"Existují tři velmi závažná omezení, která aktuálně limitují použitelnost videa v HTML5.
Za prvé, video v HTML5 neobsahuje žádná opatření ke streamování videosouborů. Uživatelé si již zvykli na to, že mohou video posouvat na vybraná místa a odtud si je prohlédnout. To je něco, v čem videopřehrávače založené na technologii Flash excelují, a to díky množství snahy, kterou společnost Adobe vložila do technologie Flash jako do platform pro dodávání videa. Abyste se mohli posouvat ve videu v HTML5, soubor se musí v prohlížeči celý stáhnout. To se ale může časem změnit.
Za druhé, zde neexistuje způsob, jímž spravovat práva. Weby, jako Hulu, které chtějí předcházet pirátství jejich obsahu, se mohou na video v HTML5 spolehnout. Flash tedy zůstava v této situaci jediným reálným řešením.
A nakonec, a to je to nejdůležitejší, process kódovanání videa je finančně a časově náročný. Kvůli potřebě kódovat video do vice formátů, je video v HTML5 ménů atraktivní.
Z toho důvodu můžete vidět mnoho webů poskytujících video v patentovaném formátu H.264, takže je možné je přehrát na iPodu a iPadu za pomoci kombinace element HTML5 video a Flashe."[1]

Kontajnery a kodeky

Kontajnery a kodeky sa spájajú s videom na webe, pričom môžeme zjednodušene povedať, že môže ísť o video vo formáte AVI alebo MPEG. Kontajnery tvoria obal obsahujúci prúdy audia,

videa a metadát, ako sú napríklad tituly. Tieto prúdy je potrebné kódovať, a o to sa starajú kodeky. Audio a video sa dá zakódovať rôznymi spôsobmi, ale ak sa ide o prácu s HTML5, dokážu to len niektoré z nich.[1]

Kodeky a podporované prehliadače

- H.264 (IE9, S4, C3, IOS)
- Theora (FF3.5, CH4, O10)
- VP8 (IE9, FF4, CH5, O10.7)

H.264

Je určený pre videa s vysokou kvalitou obrazu. Špecifikácia je rozdelená do niekoľkých profilov tak, aby podporovala lacnejšie zariadenia, ale aj zariadenia s vysokým rozlíšením. Video je možné zakódovať naraz a potom vložiť do viacerých profilov, takže bude mať dobrú kvalitu na viacerých platformách. Ide o štandard, pretože podporuje Microsoft aj Apple. Video môže byť kódované týmto kodekom až po zaplatení licencie.[1]

Theora

Ide o voľne dostupný kodek vyvinutý Xiph.Org Foundation. Tvorcovia dokážu vytvoriť video s podobnou kvalitou ako H.264, ale adaptácia prebieha pomaly. Firefox, Chrome a Opera dokážu prehrávať takéto videá bez prídavného softvéru, ale ostatné ich prehrávať nedokážu.[1]

VP8

Kodek od Googlu, ktorý je voľný a nevyžaduje žiadne poplatky, má podobnú kvalitu ako H.264. Podporuje ju Mozzila, Google Chrome, Opera a dokonca i Adobe Flash Player, čím sa stáva vhodnou alternatívou.[1]

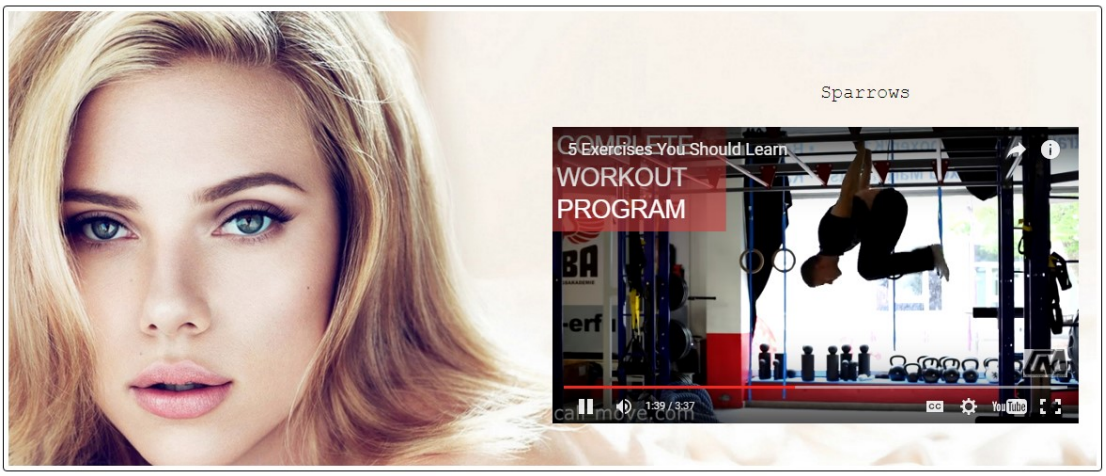
V našom prípade sme sa snažili pridať video do SVG canvasu, pričom sme dospeli k záveru, že sa to nebude dať priamo. Jediná možnosť ako pridať video sa javila cez iframe, ale na riešenie sme neprišli. Iframe sa do D3 canvasu pridáva prostredníctvom objektu typu foregein object. Po dlhom testovaní sme na programátorskom fóre dostali odpoveď, že sa video nedá pridať do canvasu, takže sme sa rozhodli, že nebudeme element video riešiť.

Jediné možné riešenie je pridanie videa mimo canvasu cez html kód a v kaskádovom štýle mu nastaviť absolútnu pozíciu, čo znamená, že sa nebude viazať na obsah stránky, a nastaviť mu pozície prostredníctvom parametrov top a left. Výsledok vizuálne zobrazuje video v canvase, ale v technickom stave sa tam nenachádza. Takýmto spôsobom však nedokážeme dostatočne zabezpečiť chovanie videa tak, aby malo pozitívny vplyv na celkový priebeh prezentácie a vyhovovalo konečnému stavu Slide Show.

```
<div id="videoAppend">
   <iframe src="https://www.youtube.com/embed/9wdWiZ7D0l8" frameborder="0"
       allowfullscreen></iframe>
</div>

<style>
   #videoAppend {
      position: absolute;
      top: 60%;
      left: 50%;
      width: 50%;
      height: 20%;
   }
</style>
```

Výpis 4: Ukážka vizuálnej implementácie elementu video do canvasu



Obrázek 4: Konečný stav videa v D3 canvase

4 Predstavenie aplikácie

Hlavnou úlohou aplikácie je vytvorenie editora, kde si bude možné pridávať vlastný obsah, efekty a vlastnú vizualizáciu. Následne sa obsah uloží do súboru, z ktorého sa budú tieto dáta využívať na zobrazenie vlastnej vizualizácie.

4.1 Použité technológie

HTML

Hypertextový značkový jazyk HTML (HyperText Markup Language) je značkový jazyk, ktorý je zameraný na vytváranie webových stránok. Kladie dôraz na prezentovanie informácií ako sú odseky, tabuľky, fonty a podobne. Značky HTML sú ohraničné v ostrých zátvorkách, sú párové a nepárové, obsahuju atribúty.[11] „HyperText“ znamená možnosť prepojenia textov na základe odkazov. „Markup“ reprezentuje schopnosť HTML určovať význam konkrétnym blokom textu pomocou špecifických značiek a tagov, čím sa tento blok vizuálne zmení a upraví. Je to napríklad verzia „tučného“ písma alebo označenie bloku ako „nadpis“ niektorej z úrovní. Modernejšia verzia tohto jazyka je označovaná ako XHTML a čoraz viac webmasterov ju začína využívať.[12]

CSS

Kaskádové štýly označujú formátovanie stránok, ktoré sú písané v HTML, XHTML alebo XML. Kaskádové štýly sa používajú pri definovaní vzhľadu web stránky, veľkosti písma, farieb a podobne. Ich hlavnou výhodou je bezpochyby samotná „kaskádovosť“. Jednotlivé štýly v CSS môžu byť vzájomne poprekrývané a nadradené či podradené ostatným. Samotný vzhľad dokumentu sa dá oddeliť od jeho obsahu, ak je CSS správne nakódované. Vznikajú tzv. beztabulkové layouty. Oddelenie týchto dvoch vrstiev – vzhľadovej a zdrojovej štrukturálnej vrstvy, zvyšuje prístupnosť samotného webu, a to je hlavný rozdiel oproti formátovaniu s atribútmi, ktoré sa používalo v minulosti.[13]

"HTML5 a CSS3 pomáhají rozvrhnout základ pri příšti generaci webových aplikací. Umožňují nám tvořit weby, které se snadněji vyvíjí a spravují, a jsou přivětivější k uživatelům. HTML5 obsahuje nové elementy definujíci štrukturu a vkládání obsahu, čoz znamená, že nemusíme pužívat zdrojové kódy navíc nebo zásuvné moduly a další doplňky. CSS3 poskytuje pokročilé selektory, grafická vylepšení a lepší podporu fontů, díki čemuž jsou naše weby vizuálně atraktivnější, aniž by bylo potřeba použít techniku nahrazování obrázků, složitý JavaScript nebo grafické nástroje. Vylepšená podpora ptistupnosti zlepší ajaxové aplikace pri lidi s omezením a off-line podpora nám umožňuje začít budovat fungující aplikace, které nevyžadují internetové připojení."[1]

Javascript

Objektovo orientovaný programovací jazyk, ktorý ma využitie pri tvorbe webových stránok. Javascript beží na strane prehliadača až do momentu stiahnutia do počítača.Tu je rozdiel medzi serverovým jazykom, ako je PHP, ktorý generuje kód webu, a spomínaným Javascriptom. Tento programovací jazyk nájdeme pri tvorbe interaktívnych webových stránok. Javascriptom sa môže docieliť napríklad: správne vypĺňanie formulárov, meniace sa obrázky a po kliknutí či podržaní myši na meniacich sa obrázkoch aj rozbaľovanie menu a mnoho ďalšieho. Možno ho tiež využiť aj ako meradlo štatistík návštevnosti. Programovací jazyk JavaScript patrí s HTML a CSS do DHTML, ktoré zlepšuje užívateľské rozhranie a celkovo je web pre návštevníka viac priateľský a dynamickejší.[14]

jQuery

jQuery je JavaScriptovská knižnica, dalo by sa povedať, že dokonca až framework, ktorý slúži na uľahčovanie práce s JavaScriptom. Prináša množstvo efektov a taktiež podporu Ajaxu. Veľkou prednosťou tejto knižnice je jednoduchosť písania príkazov a najmä ich reťazenie.[15]

Ajax

Asynchronous JavaScript and XML. AJAX nie je novou technológiou, ale iba novou kombináciou technológií, ktoré sú už znám, ánapríklad HTML (alebo XHTML), JavaScriptu, XML. Aplikácie využivajúce AJAX dokážu odoslať a získat data zo serveru bez nutnosti znovunahrávania celej stránky. AJAX tak môže mať veľké využitie, príkladom sú našepkávače (formuláre, ktoré se automaticky predvyplňujú podľa stlačenej klávesy), ankety a ďalšie zložitejšie aplikácie, ktoré dokážu uživateľom uľahčiť prácu.[16]

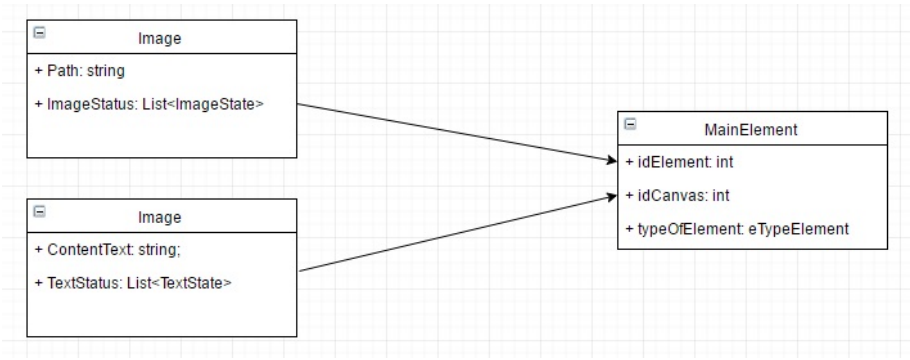
ASP.NET

ASP.NET je jednotný model pre vývoj webu, ktorý zahŕňa služby potrebné na tvorbu webovej aplikácie s minimalizovaním kódovania. Je súčasťou rozhrania .NET Framework. Pri vývoji je dostupný ľubovoľný programovací jazyk, ktorý je kompatibilný s CLR (Common Language Runtime), hlavne Microsoft Visual Basic a C#. Tieto jazyky umožňujú rozvíjať aplikácie ASP.NET, ktoré ťažia Common Language Runtime, dedičnosti a podobne.[17]

Pre vývoj aplikácie sme si vybrali programovací jazyk C#, nakoľko mám s ním najviac skúsenosti a spĺňa všetky podmienky pre prácu s ASP. Umožňuje vývojárom vytvárať bezpečné a robustné aplikácie, ktoré pracujú na .NET Framework. Môže sa použiť pre tvorbu aplikácií klienta systému Windows, webové služby XML, aplikácie typu klient-server, databázové aplikácie

a podobne.

4.2 Prehľad a realizácia efektov



Obrázek 5: Dátova vrstva aplikácie

4.3 Popis jednotlivých modelov aplikácie

Main Element

Ide o hlavný model, od ktorého sa budú odvíjať ostatné elementy. Keďže každý element má svoje vlastné vlastnosti, vznikla potreba oddeliť ich od seba. V modeli MainElement, ktorý reprezentuje základnú triedu, sa nachádzajú všetky vlastnosti, ktoré majú všetky elementy spoločné. Všetky prvky budú dediť od tohto modelu – týmto dosiahneme jednotnosť prvkov na canvase ako prvky typu MainElement.

Image

Obrázok pre správne fungovanie potrebuje mať svoje osobité vlastnosti. Keďže sa bude v určitý čas a v určitom mieste meniť, vznikla potreba tieto stavy zachytávať a zaznamenávať. Tieto stavy sa budú určovať v editori aplikácie po potvrdení užívateľom. Model stavu obrázku bude prezentovaný neskôr v tejto časti. Pri obrázku nastáva problém pri zmene veľkosti, kedy sa nedokážu správne prepočítať súradnice, preto sa odporúča používať efekt zväčšenia.

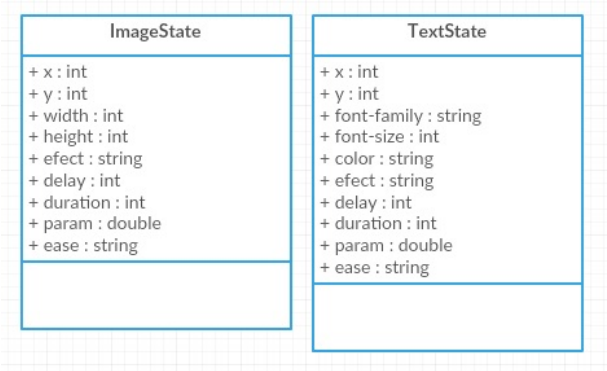
Text

Aby sa text mohol dynamicky meniť, potrebuje taktiež svoje vlastné vlastnosti. Stav obrázku a textu je v určitý čas rôzny, preto sa musia aj tieto stavy zaznamenávať. Dôvodom na rozdelenie

stavu obrázku a textu je, že text má iné vlastnosti, ako napríklad farba, typ, font písma. Oproti tomu má obrázok výšku a šírku.

Video

Dynamické video nemá veľké využitie, preto je zbytočné, aby sa prehrávané video dynamicky menilo. Môže meniť svoju pozíciu, ale efekty ako rotovanie, zvrtenie a podobne podporovať nebude. Video v tejto práci riešné nebude, ale pre potreby v budúcnosti by sa využila len pozícia videa bez špeciálnych efektov.

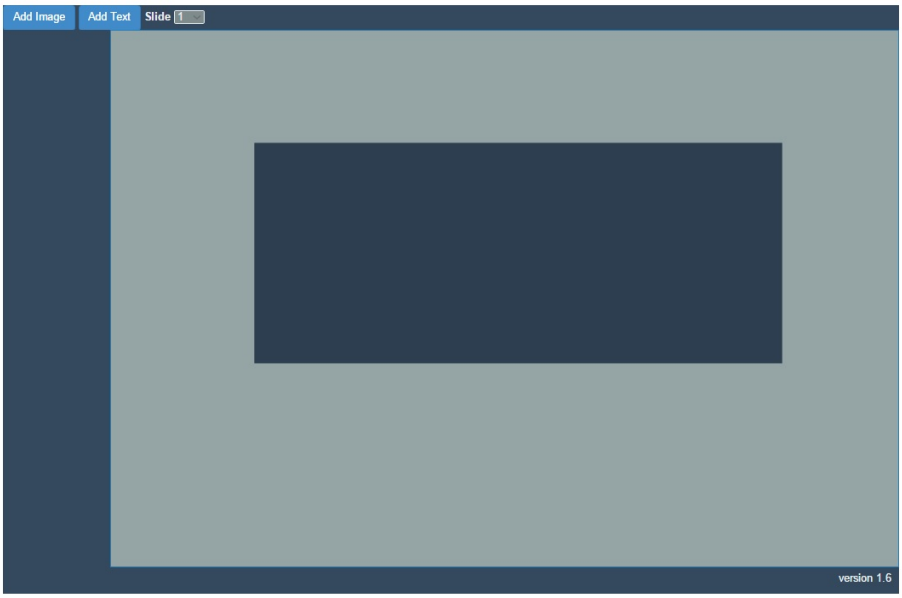


Obrázek 6: Stavy elementov

5 Editor vizualizácie

Editor je užívateľské rozhranie, kde sa budú do canvasu pridávať elementy. Ďalej sa bude s nimi pracovať už priamo v canvase. Dizajn je navrhnutý tak, aby mal užívateľ plnú kontrolu nad elementmi a s ich spojenou tvorbou vizualizácie. Layout je rozdelený do štyroch hlavných častí. Každý editor musí spĺňovať tri základné vlastnosti – musí byť jednoduchý na ovládanie, musí pracovať na základe požiadaviek užívateľa a musí byť z rozloženia jasné, kde sa ktoré vlastnosti a funkcie budú nachádzať. Tieto normy slúžia na vytváranie produktu, ktorý bude mať predpoklady na ďalšie používanie. Príklady editoru môžu byť: skicár, MS Power Point, MS Office a podobné produkty.

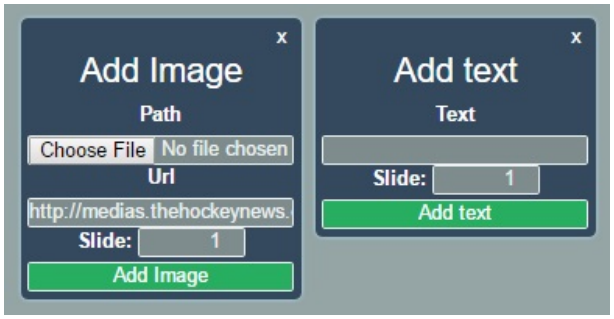
Editačný editor musí zachytávať zmeny a tieto zmeny realizovať podľa potreby alebo v časovej odozve. Mal by zobrazovať aktuálny stav a jeho možné budúce vlastnosti. Podľa potreby je možné jednotlivé zmeny buď meniť alebo mazať. Dizajn by nemal obsahovať veľa farebných schém, pretože by pôsobil nepohodlne. Musí sa dodržiavať účel, na ktorý je určený.



Obrázek 7: Rozloženie layoutu editora

Horný panel, hlavička editora, obsahuje tlačidlá pre pridávanie elementov textu a obrázkov do canvasu. Má v sebe implementované skryté formuláre pre vstupné informácie elementu. V prípade pridávania textu sa zobrazí okno, kde je potrebné zadať obsah textu s číslom snímky, kde sa bude zobrazovať, a následne potvrdiť. Pri pridávaní sa automaticky načíta zo zdroja defaultná šírka a výška v pixeloch. V prípade zamietnutia potvrdenia sa okno skryje a objekt nebude pridaný. V opačnom prípade sa zobrazí element v canvase editoru, kde bude naďalej dostupný k editácií. Jednotlivé snímky sa presúvajú prostredníctvom výberu, ktorý sa reprezentuje

zoznamom snímok. Po zmene sa zobrazí slide so svojimi elementmi a ostatné, ktoré nepatria do aktuálnej snímky, nebudú dostupné. Každý element je dostupný iba na svojej snímke, tým sa zabráni miešaniu slidov a prvkov.



Obrázek 8: Ukážka vstupných formulárov

Problém môže nastať, keď sa užívateľ snaží pridať obrázok vo vysokej kvalite s vysokým rozlíšením. Po potvrdení sa formulár ukončí úspešne, ale objekt v prostredí canvasu neuvidíme. Riešenie je dosiahnuté stanovením limitu rozlíšenia obrázku. Ak sa pridáva obrázok ako pozadie pre celý slide a chceme, aby sa aj toto pozadie dynamicky menilo, treba myslieť na to, aby boli rozmery väčšie ako rozmery slidu, pretože by sa mohli objaviť na slide výrezy.

V ľavej časti editora sa nachádza editačné menu pre elementy. Keďže sa elementy rozlišujú podľa ich typu, toto menu sa bude meniť. Pred pridaním objektu je editačné menu prázdne a zobrazí sa až po kliknutí na element. Pohyb prvkov po canvase sa realizuje ťahaním myškou po canvase (funkcia drag). Po zobrazení ponuky sú na výber vlastnosti a efekt, ktorý pozostáva z kľúčového slova, ktoré sa vyberá z drop down listu. Každý efekt pozostáva z oneskorenia a z trvania, čím sa získava vlastná časová nezávislosť. Efekty prebiehajú postupne tak, ako si ich užívateľ navolí. Pre ukážku elementu a efektu pred potvrdením slúži tlačidlo Save, ktoré zrealizuje efekt a vráti ho naspäť na pozíciu. Po každom potvrdení sa elementy aktualizujú a ukladajú sa v serverovej časti do špeciálneho XML súboru. Táto funkcia ukladania a aktualizácia pozostávajú z vytvorenia JSON string objektu, ktorý obsahuje všetky potrebné vlastnosti a predá tieto vlastnosti do serverovej časti aplikácie, konkrétne do CODE BEHIND, ako parameter funkcie. Takýto objekt sa vytvorí z preddefinovaného typu, do ktorého sa uložia hodnoty a následne sa konvertuje z objektu do JSON stringu. V prípade potreby trvalého vymazania objektu z canvasu je k dispozícií tlačidlo delete, ktoré vymaže prvok z canvasu, ale aj zo serverovej časti, konkrétne z XML súboru. Každý prvok je jedinečný, a preto sa musí odlišovať. Pri vkladaní sa pridelí jedinečné identifikačné číslo, pomocou ktorého sa bude realizovať konkrétne vyhľadávanie prvku alebo jeho mazanie. Vyhľadávanie sa realizuje buď využitím práve tohto identifikačného prvku alebo vyberaním kolekcie pomocou typu prvku. Typy sa zadávajú pri pridávaní, kedy sa kľúčovým slovom označí, či sa pridá text alebo obrázok. Ďalšou možnosťou je vybranie podľa hodnoty atribútu. Výsledkom bude kolekcia, ktorá bude vyhovovať podmienke vyhľadávania.

V spodnej časti editora sa nachádza päta, ktorá je nositeľom aktuálnej verzie, slúži len na informačné účely.

Obrázek 9: Vlastnosti editora pre element text (vľavo) a obrázok (vpravo)

```
var Img = new ImageState(...);

$.ajax({
      type: 'POST',
      url: '/About.aspx/SetImageStatus',
      dataType: 'json',
      contentType: 'application/json; charset=utf-8',
      data: JSON.stringify({
            imgData: Img
            }),
      success: function (response) {
            aler("success");
           },
      error: function (error) {
            alert(error.statusText);
            console.log(error);
            }
});
```

Výpis 5: Posielanie dát do systému

5.1 Vizuálne efekty

Táto podkapitola je zameraná na riešené efekty, ktoré sme v práci realizovali. Efektom sa myslí zmena stavu objektu, a to so špeciálnou vizuálnou vlastnosťou, ktorou zmení svoje chovanie. Vďaka týmto efektom sa tvoria dynamické prvky, ktoré sa definujú v kaskádovom štýle, v javascipte alebo pomocou D3.js, ktorý dokáže komunikovať so štýlmi a meniť im vlastnosti za behu programu na strane klienta. Efekty pre jednotlivé prvky sa trochu líšia pri realizácii. Z tohto dôvodu sme sa rozhodni venovať tomu danú podkapitolu.

Akákoľvek technika na vytvorenie obrázkov, animácie alebo ďalšieho grafického prvku, ktorá sa podľa určitých výpočtov a zrealizovaných funkcií zobrazí do grafickej podoby na displeji, sa nazýva vizualizácia. Vizualiácia môže byť statická alebo dynamická. Statická vizualizácia sa skladá z elementov, ktoré nijak nereagujú na vonkajšie dianie a nemenia svoj charakter a stav. Naopak, dynamická vizualizácia reaguje na podnety vyvolané užívateľom, časovým harmonogramom alebo na podnety špeciálnych reakcií na stav systému. Pre webové aplikácie je dynamická vizualizácia prínosnejšia, pretože dokáže rýchlejšie upútať pozornosť a reakcie. S dynamickosťou treba pracovať opatrne, aby neriešila zbytočne veľa prvkov a vyhovovala ostatným vlastnostiam stránky, ako môže byť napríklad responzívny dizajn stránky.

Responsibilitou stránky sa myslí stajlovanie stránky tak, aby zobrazenie bolo optimalizované pre rôzne druhy zariadení, ako sú notebook, tablet, smartphone, mobily. Má tri základné úrovne – flexibilná štruktúra, flexibilné obrázky a Media Querries.[18]

Flexibilná štruktúra

Realizuje sa v percentuálnom zadávaní šírky, nie pomocou pixelov. Takto môže reagovať na rôzne šírky displeja na zariadení. Výpočet sa realizuje pomocou nasledujúceho vzorca. Kontext je v roli obalového tagu, ktorý má šírku 500px a potrebujeme vložiť blok o šírke 250px. Šírku v percentách vyjadríme nasledovne: Takto dosiahneme 50-percentné zaplnenie obalového kontextu pridávaným blokom, ktoré pri zmenšení bude zapĺňať stále 50-percentný obsah.[18]
percentuálna šírka = (požadovaná šírka (v pixloch) / kontext (v pixloch)) * 100

Flexibilné obrázky

Proces prispôsobuje obrázok samotnej štruktúry, realizuje sa to tak, že sa pri pridávaní obrázku cez tagy <img> neudáva šírka ani výška obrázku. Tá sa nastavuje kaskádovým štýlom cez atribúty. Max-width sa nastaví na 100% a height sa nastaví na auto.[18]

Media Queries

"Už nějakou dobu jsme schopni definovat šablonu stylů v závislosti na typu požitého média, ale byli jsme omezeni typem výstupu, jak jste mohli vidět v části Tvorba tisknutelných odkazů pomocí: after a content na straně 89, kde jsme definovali naši tiskovou šablonu stylů. Media Queries (česky dotazovaní na výstupní médium) v CSS3 nám umožňují změnit prezentaci stránky na základě velikosti obrazovky, kterou používa návštevník webu. Weboví vývojáři prováděli detekci velikosti obrazovky roky pomocí JavaScriptu, aby získali informaci o velikosti obrazovky uživatele. Nyní můžeme totéž začít provádět pomocí samotné šablony stylů. Takže dotaz na médium můžeme použít k určení:

- *Rozlišení*
- *Orientace (na šírku nebo na výšku)*
- *Šířky a výšky zařízení*
- *Šířky a výšky okna prohlížeče*

Díky tomu nám dotazy na médium zjednodušují tvorbu šablon alternativních stylů určených mobilním užívatelům."[1]

5.2 Realizácia vizuálnych efektov

Všetky vizuálne efekty v D3.js sa riešia prostredníctvom pridania funkcie transition k elementu. Pridaním transition k elementu definujeme, že v objekte nastane zmena v jeho chovaní, to znamená, že môže zmeniť svoju polohu so špeciálnym vizuálnym efektom. Môže sa premiestniť v canvase, zrotovať, skosiť a podobne. Ak chceme, aby element mal viac efektov, ktoré idú lineárne za sebou, vyriešime to jednoducho – pridáme ďalší efekt, znova pridaním funkcie transition k elementu. Bez určenia časových razítok pre oneskorenia a trvania sa zmeny prevedú rýchlo a nebudeme schopní ich zaregistrovať. Práve preto pridaním hodnoty duration pre trvanie a delay pre oneskorenie budeme schopní zmeny postrehnúť. Hodnoty sa zadávajú v milisekundách (1 sekunda = 1000 milisekund). Problém nastal vtedy, keď sme sa snažili prehrávať efekty objektu lineárne, teda hneď za sebou s využitím časových razítok. Časové razítko pre oneskorenie sa nezväčšuje automaticky, ako by sa mohlo zdať. Nasledujúci efekt začína, keď sa vykoná oneskorenie aj trvanie predchádzajúceho. Takže nasledujúci čas pre štart efektu dostaneme súčtom časových razítok. Pri pridaní ďalších efektov sme museli zakaždým prepočítavať oneskorenie súčtom celkového trvania a aktuálneho oneskorenia.

Translácia (Presun)

Pre využívanie efektov sme museli správne pridávať elementy do canvasu a určiť im ich pozíciu. Sú na to dva spôsoby – definovaním atribútov „x“ a „y“ alebo presunom pomocou premiestnenia na pozíciu. Výber je diskutabilný a závisí od požiadavky. Keď sme definovali pozíciu

prostredníctvom definovania jeho súradníc pomocou atribútov „x“ a „y“ a použili sme na premiestnenie presun, nedostali sme presun na požadovanú pozíciu, ale posun po x-ovej a y-ovej súradnici o „x“ a „y“ pixeloch, a nie na túto súradnicu. Keď sme použili opäť definovanie súradníc, výsledok fungoval správne. Ak sme využili druhú metódu, zadali sme začiatočnú pozíciu presunom na pozíciu a pre ďalšie presuny sme taktiež využili presun na pozíciu. Takto všetko fungovalo správne, ale ak sme pri tomto definovaní nevyužili presun na pozíciu, ale vyplnenie atribútov „x“ a „y“, dostali sme rovnakú chybu, ako pri prvom definovaní. Dospeli sme k záveru, že nebudeme miešať definovanie súradníc s presunom na pozíciu, ale budeme využívať len jednu metódu, čím sa vyhneme zbytočnému prepočítavaniu.

```
canvas.append("text")
   .text("Spravne riesenie")
   .attr("font-size", 40)
   .attr("text-anchor","middle")
   .style("fill","green")
   .attr("transform", "translate(20,20) ")
   .transition()
   .attr("transform", "translate(200,150)")
   .duration(2000)
   .delay(1000);

canvas.append("text")
   .text("Nespravne riesenie")
   .attr("font-size", 20)
   .style("fill","red")
   .attr("text-anchor","middle")
   .attr("x", "20")
   .attr("y", "20")
   .transition()
   .attr("transform", "translate(200,200)")
   .duration(2000)
   .delay(1000);
```

Výpis 6: Ukážka práce s transáciou

Zväčšenie (Scale)

Ide o typ efektu, ktorý zväčší objekt v merítku podľa veľkosti prvku. Obsahuje jeden parameter, ktorým charakterizujeme koľkonásobne sa prvok zväčší. Testovaním sme zistili, že pri

rozdielnom definovaní sa elementy chovajú inak. Ak definujeme elementy pomocou jeho atribútov „x“ a „y“, merítko sa nastavuje samostatne. Ak budeme mať element na určitej pozícii a budeme naň aplikovať tento efekt o dvojnásobné zväčšenie, tak sa nám posunie o dvakrát ďalej, ako má svoje pozície, a tam sa ešte zväčší na dvojnásobný objem, čiže nejde o komplexné riešenie. Naopak, ak je prvok definovaný posunom na začiatočnú pozíciu a aplikujeme naň efekt zväčšenia, tak sa na definovanej pozícii zväčší bez toho, aby sa posúval.

```
image1.transition()
   .attr("transform", "translate(50,50) scale(2)")
   .duration(2000)
   .delay(1000);

image2.transition()
   .attr("x", "100")
   .attr("y", "50")
   .attr("transform","scale(0.5)")
   .duration(2000)
   .delay(1000);
```

Výpis 7: Ukážka práce so zväčšením

Text sa zväčšuje v mieste, ktoré je nastavené v atribúte „text-anchor“. Môže sa zväčšovať rovnomerne zo stredu, z ľavej alebo z pravej strany. Naopak, obrázok sa zväčšuje iba z ľavého horného rohu, čo znamená, že sa bude zväčšovať iba z pravej strany.

Viditeľnosť (Opacity)

Je to efekt, s ktorým sa nastavuje, či je element viditeľný v canvase alebo je skrytý. Jeho funkciou nie je vymazanie objektu, ale iba jeho viditeľnosť. Pri voľbe tohto efektu treba brať na vedomie, že ak chceme aby sa viditeľnosť objektu zmenila na skrytý, volá sa parameter 0. V opačnom prípade sa volá parameter 1. Je povolené voliť hodnoty v rozmedzí od 0 do 1, čo charakterizuje 0% až 100% viditeľnosť.

```
image1.transition()
   .attr("transform", "translate(150,50)")
   .style("opacity",0.25)
   .duration(2000)
   .delay(1000);

image2.transition()
```

```
    .attr("x", "150")
    .attr("y", "200")
    .style("opacity",0.75)
    .duration(2000)
    .delay(1000);
```

Výpis 8: Ukážka práce s veiditeľnosťou

Skosenie (Skew)

Má za úlohu skosiť element podľa osi x alebo y. Skosením sa myslí ťahanie hrany, ktorá je rovnobežná k osi, podľa ktorej sa vykonáva kosenie. Hrany sa ťahajú v protismere. Problém nastáva pri zobrazení kosenia v uhloch 90°a 270°, nakoľko sa strany navzájom prekrývajú, čo má za následok, že objekt nie je viditeľný.

```
image1.transition()
    .attr("transform", "translate(150,150) skewX(40)")
    .duration(2000)
    .delay(1000);

image2.transition()
    .attr("x", "150")
    .attr("y", "150")
    .style("opacity",0.50)
    .attr("transform","skewY(40)")
    .duration(2000)
    .delay(1000);
```

Výpis 9: Ukážka práce s kosením

Testovaním sme zistili, že po použití efektu kosenia nefunguje správne ďalší presun ani ostatné efekty. Dôvodom je zlá orientácia pozície, pretože pri kosení sa hrany posúvajú a menia súradnice elementu. Dospeli sme k riešeniu, že zaistíme normalizáciu elementu. Dostaneme ju tak, že po aplikovaní skosenia v určitom uhle musíme pri ďalšom aplikovaní efektu nastaviť skosenie pri uhle 0, čím dostaneme normalizovaný element. S takýmto elementom sa dá ďalej korektne pracovať.
Normalizovanie sa musí zaistiť pri elementoch, ktoré sa definujú atribútmi pozície „x“ a „y“. Ak je objekt definovaný posunom už od začiatku, tak sa normalizácia nerieši, pretože funkcia posun na pozíciu ju rieši samostatne.

Rotácia (Rotate)

Rotovanie elementu slúži na zrotovanie v určitom uhle. Pri obrázku sa realizuje v ľavom hornom rohu, avšak pri texte sa rotuje podľa nastavenia atribútu „text-anchor“, čiže v ľavom alebo v pravom rohu, prípadne v strede textu.

Pri definovaní elementu pomocou atribútov „x“ a „y“ nefunguje rotácia správne, pretože sa rotuje okolo ľavého horného rohu canvasu, a nie po svojej osi. Po aktualizácii súradníc sa osi x a y zrotujú v uhle objektu, a nie podľa canvasu. To znamená, že ak sa rotuje o 90°, vymenia sa osi x a y, pričom vzniká ďalší problém. Testovaním sme našli riešenie. Je nutné zadať súradnice pri rotovaní ako parameter hneď za uhlom a následne aktualizovať s rovnakými hodnotami aj parametre „x“ a „y“.

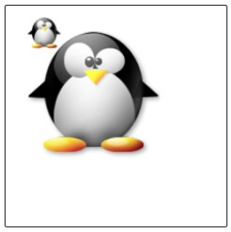
```
 image1.transition()
   .attr("transform", "translate(150,150) rotate(90)")
   .style("opacity",0.25)
   .duration(2000)
   .delay(1000);

image2.transition()
   .attr("transform", "rotate(90,150,300)")
   .attr("x", "150")
   .attr("y", "300")
   .style("opacity",0.75)
   .duration(2000)
   .delay(1000);
```

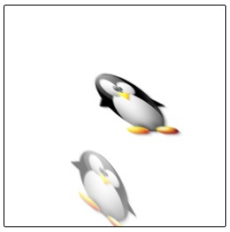
Výpis 10: Ukážka práce s rotovaním

Pri ďalšom testovaní sme zistili, že objekt sa nám pri definovaní „x“ a „y“ zrotuje správne, ale po pridaní ďalšieho efektu sú osi stále otočené v uhle rotácie. Vzniká tu potreba ďalšej normalizácie, ktorú dostaneme zrotovaním elementu na uhol 0°, avšak už bez ďalších povinných parametrov.

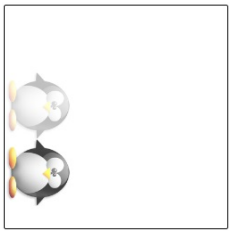
Obrázek 10: Výsledok translácie



Obrázek 11: Výsledok zväčšenia



Obrázek 12: Výsledok skosenia



Obrázek 13: Výsledok rotácie

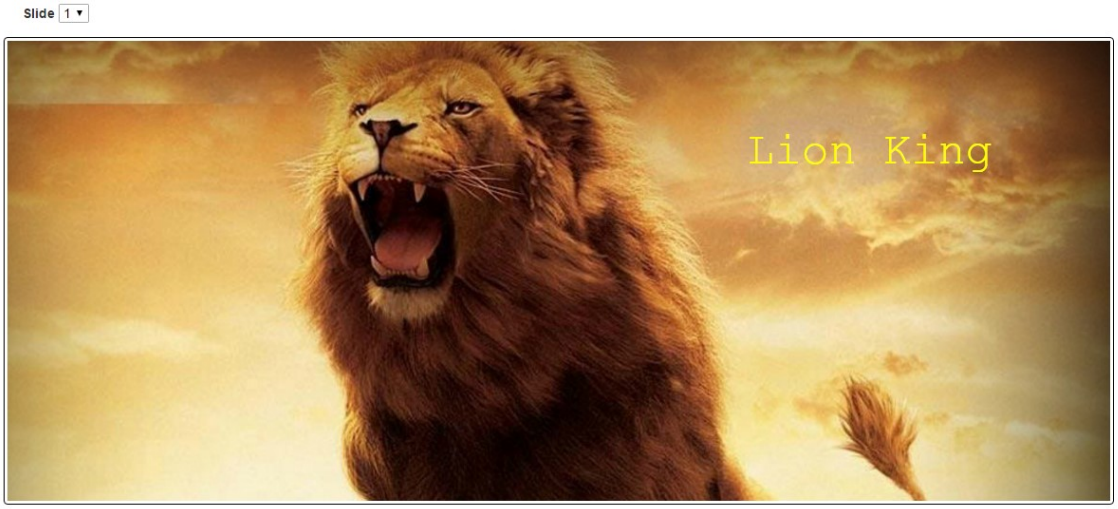
5.3 Výber techniky vizualizácie

Výber techniky pre editor a vizualizáciu Slide Show je zhodný, pretože tak, ako sme ukázali v predchádzajúcej kapitole, nie je dobré a odporúčané kombinovať jednotlivé typy definovania pozície elementov. Pre editor sme sa rozhodli definovať element cez presun na pozíciu. Dôvodom je, že v editore nebude za sebou riešených viacero efektov. Editor ponúka prehľad možnej vizualizácie, avšak pre ukážku dostatočne stačí zobraziť vybraný efekt, ktorý sa bude dať ľahko vrátiť do predchádzajúceho stavu, pretože sa bude dať ľahko normalizovať pre ďalšie použitie, túto normalizáciu si posun rieši sám.

Pre prezentáciu Slide Show sme vybrali definovanie pozície elementov prostredníctvom posunu na požadovanú pozíciu, nakoľko si táto metóda normalizáciu rieši sama a eliminujeme vznikanie elementov, ktoré budú potrebovať ručnú normalizáciu. Nevýhoda tejto metódy je však v tom, že z hľadiska respozivity nemôžeme riešiť pozíciu bootstrapom v zadávaní hodnôt v percentách. D3 nepodporuje v definovaní atribútu „translate“ percentá. Riešenie tejto nevýhody sme realizovali pomocou zisťovania dĺžky a šírky obaľovaného prvku. Vysvetlenie bude prezentované v kapitole číslo 6 a podkapitole číslo 1.

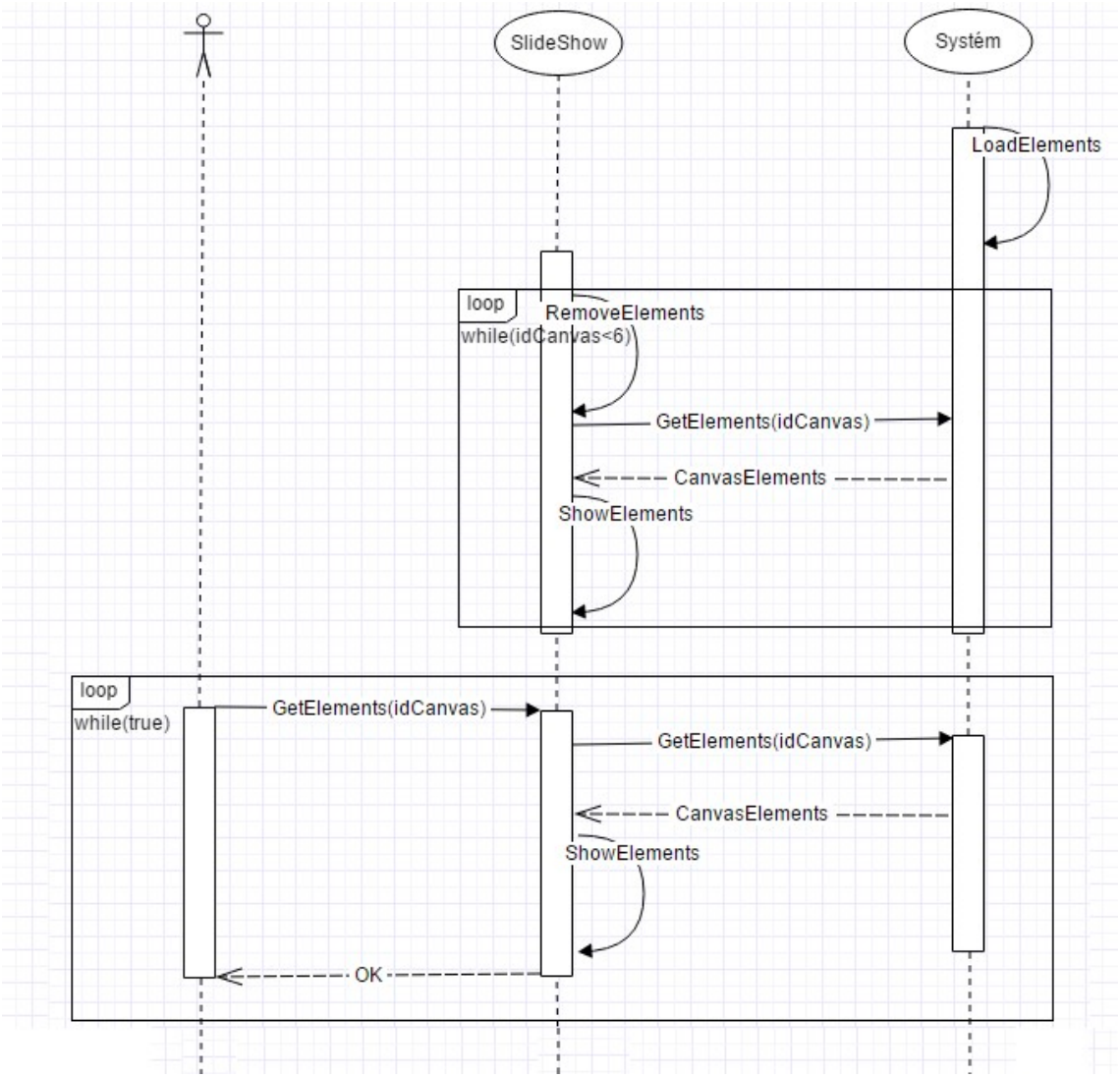
6 Vizualizácia Slide Show

Jednotlivé slidy sa realizujú do D3 canvasu. Najskôr sa v systéme načítajú všetky elementy z XML súboru, v ktorom sú uložené pre každý element všetky jeho stavy. Podľa svojho jedinečného identifikačného čísla sa zoradí množina elementov vzostupne. Toto identifikačné číslo slúži na rozdelenie prvkov na vrzstvy. Každý objekt canvasu má práve jednu vrzstvu. V algoritme to znamená, že sa vkladanie začína od najmenšieho identifikačného čísla. Element s vyšším identifikačným číslom má vačšiu prioritu v zobrazovaní v canvase a prekrýva element s nižžším identifikačným číslom. Tento jedinečný identifikátor sa nemení ani po pridaní efektu k prvku. V množine všetkých elementov sa identifikačné čísla prvkov nezhodujú ani pri rôzných snímkach. Prvý uložený stav slúži ako štartovacia pozícia. Každý element sa bude zobrazovať cez efekt viditeľnosti (Opacity)a funkciou presunu na pozíciu, čím sa vyhneme normalizácií elementov, pretože táto funkcia si jú rieši sama.



Obrázek 14: Ukážka slideru

Zobrazovací canvas je na začiatku prázdny. Najskôr sa získajú elementy prvého canvasu a následne sa zobrazia. Pri definovaní sa v rovnakom čase definujú aj všetky jeho stavy s efektom. Môžeme to takto realizovať, pretože časová náročnosť definovania elementu so všetkými stavmi je zanedbateľná a neovplivňuje priebeh vizualizácie. Po definovaní sa začnú elementy zobrazovať. Každý element je samostatný, pretože sa nachádza na svojej vlastnej vrstve. Vďaka vrstveniu môžeme konfigurovať priority zobrazovania objektov v canvase.
Po dokončení zobrazenia slidu sa všetky objekty v canvase vymažu. Následne sa získavajú zo systému všetky elementy nasledujúceho slidu, ktorý je určeny jeho identifikačným číslom. Toto číslo je pre každý slide jedinečné. Zobrazovací priebeh prebieha stále rovnakým priebehom. Po zobrazení poslednej snímky sa prezentovanie zastaví. Automatické prehrávanie snímok dokážeme realizovať pomocou implementovania časového razítka, ktoré je porovnávané s celkovým časom prezentovania aktuálnej snímky.



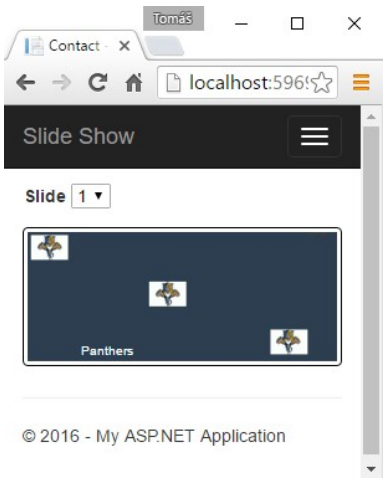
Obrázek 15: Sekvenčný diagram fungovania procesu Slide Show

6.1 Responzívny dizajn

Aplikácia je plne responzívna, ale je potreba po resize okna spustit prezentáciu od začiatku. D3 síce podporuje nastavovanie pozícií a určitých vlastností cez percentá, ale keďže sme využili techniku realizácie definovania atributov a pohybu po canvase presunom na určenú pozíciu, nie je možné využitie percentuálneho vyjadrenia polohy. Hlavným dôvodom je aj riešenie vizualizácie tak, aby sa techniky definovania atribútmi pozície a presunom na pozíciu nemiešali. Tento problém je rozoberaný v kapitole číslo 5 a podkapitole číslo 2. Týmto je zabezpečený plynulý priebeh vizualizácie. Aby užívateľ nemusel ručne štartovať vizualizáciu od začiatku, aplikácia si to vyrieši sama. Riešením je zobrazenie aktuálnej zobrazenej snímky od začiatku. Týmto zabezpečíme normalizáciu a plynulosť prebehu dynamickej vizualizácie. Výhodou je, že nemusíme odznova prepočítavať a meniť polohu elementov. Tu vznikal problém, ktorý zapríčiňoval ne-

dostatočnú vizualizáciu. Keď sa definuje poloha prvku jeho štartovacej alebo efektovej pozície, nadefinuje sa všetko správne. Následne keď počas priebehu efektu začneme manipulovať s veľkosťou okna, začnú sa prepočitávať súradnice každého elementu. Ak tento objekt je v priebehu zobrazovania efektu, prepočítaním sa mu zmenia súradnice na posledne definovanú polohu a prvok sa nachádza na poslednej definovanej pozícií.

Na vysvetlenie postačí nasledujúci príklad. Definujeme prvok presunom. Následne je naň aplikovaných päť po sebe idúcich efektov. Po definovaní sa začne vizualizácia, ktorá prebieha postupne, teda lineárne. Počas zobrazovania tretej vizualizácie sa zmení veľkosť okna, začnú sa prepočítavať súradnice podľa poslednej definovanej polohy, čiže piateho efektu. Týmto sa prvok zmení na pozíciu piateho efektu a vizualizácia sa dokončí nesprávne. Preto si aplikácia tento nedostatok vyrieši sama a to načítaním aktuálneho snímku od začiatku s úpravou canvasu a elementov na aktuálne rozmery okna, čím sa normalizuje Slide Show a priebeh vizualizácie je správny.



Obrázek 16: Responzívny dizajn stránky

7 Testovanie aplikácie

Internet Explorer 9

Nedokáže správne rozložiť layout editora, nepridáva obrázky do canvasu. Nenačítava dobre ani rozmery prezentačného canvasu pre slide show. Nepodporuje výber farby cez vstup definovaný ako farba, spracováva len hexadecimálnu reprezentáciu farby. Defaultnú farbu písma nastavuje bledomodrú. Nepodporuje ani autoinkrementáciu pri vstupe typu celé číslo. Nevie správne spracovať niektoré definované farby, hlavne pri výbere z listu. Síce nepridáva obrázky, nenačíta správne veľkosť canvasu, ale na druhej strane vďaka responzivite aplikácie prezentácia funguje správne, efekty fungujú taktiež správne. Podporuje aj implementovanú funkciu pre automatickú vizualizáciu. Nie je odporúčané používať tento prehliadač na využívanie tejto aplikácie. Responzivita má najväčší problém so zobrazením HD obrázkov s vysokým rozlíšením.

Microsoft Edge

Taktiež nedokáže pridávať obrázky do editora. Text zarovnáva sprava, čiže všetky efekty sa riadia podľa pravého rohu, čím je aj odlišné realizovanie efektov. Nepodporuje vstup prostredníctvom color pickeru, zobrazuje len hexadecimálne vyjadrenie farby. Nepodporuje automatickú inrekentáciu vstupu typu celé číslo. Ako jediný prehliadač nedokáže spracovať číselný box ako veľkosť písma pre uloženie do systému. Efekty aj responzivita prebiehajú v poriadku. Dokáže správne rozložiť layout editoru aj prezentačného canvasu. Responzivita má veľký problém so zobrazením HD obrázkov s vysokým rozlíšením.

Opera

Tento prehliadač dokáže správne rozložiť layout editoru aj prezentačného canvasu. Správne pridáva obrázky, efekty prebiehajú v poriadku. Podporuje vstup farby pomocou color pickeru, správne zobrazuje atribúty v editori. Podporuje automatickú inkrementáciu vstupu typu celé číslo. Táto funkcia sa zobrazí až po pohybe myškou na daný box. Automatické prehrávanie pracuje správne. Pri responzivite sa okno nezmenší úplne. Responzivita má trochu problém pri obrázkoch s HD rozlíšením, nedokáže ho v určitých situáciách rozložiť správne na ploche.

Opera začala mať problémy pri zobrazovaní elementov pri počte 8000, kedy efekty sekali a načítavanie trvalo až 4.6 sekundy.

Firefox

Layout editora sa tu dokáže rozložiť najlepšie, neobsahuje žiadne milimetrové výseky. Krajšie zobrazenie intervalov pomocou guličiek. Automatická inkrementácia je viditeľná už od začiatku. Správne vypĺňa atribúty v editori. Podporuje vstup typu farba pomocou color pickeru. Efekty, responzivita aj automatické prehrávanie prebiehajú v poriadku. Responzivita prebieha taktiež v poriadku. Responzivita má taktiež problém so zobrazením HD obrázkov s vysokým rozlíšením.

Opera začala mať problémy pri zobrazovaní 4000 kusov, efekty sekali a načítavanie trvalo okolo 2.5 sekundy.

Google Chrome

Tento prehliadač od spoločnosti Google má najlepšie vlastnosti pre prácu s touto aplikáciou, pretože má v layoute milimetrový výsek. Čo sa týka funkcionality, pracuje najlepšie. Správne vypĺňa atribúty v editore, správne vizualizuje efekty, responzivita aj autoprehrávanie pracujú správne. Responzivita má takisto problém so zobrazením HD obrázkov s vysokým rozlíšením. Tento webový prehliadač sa odporúča na prácu s aplikáciou Dynamic Slide Show.

Chrome dokáže spracovať najviac elementov a načítavanie mú trvá najmenej. Efekty prestali pracovať korektne pri zobrazovaní až 10000 elementov, kedy načítavanie trvalo 5.9 sekundy.

8 Záver

Cieľom bakalárskej práce bolo na základe získaných teoretických znalostí vytvoriť unikátnu webovú aplikáciu, ktorá bude prezentovať dynamickú prezentáciu so špeciálnym obsahom textu a obrázkov. Po získaní prehľadu a vedomostí v danej problematike sme začali analyzovať možné riešenia, ktoré sú prezentované v jednotlivých kapitolách. Hlavnou úlohu tejto bakalárskej práce nie je práca s editorom pre vizualizáciu, ale samotná vizualizácia, ako konečnej množiny snímkov a elementov obsiahnutých v prezentačnom canvase. Webová aplikácia je tvorená z troch záložiek, ktoré majú svoj špecifický charakter. Prvá záložka s názvom Introduction predstavuje základné pokyny a pravidlá pre úspešnú a komplexnú prácu s editorom tak, aby výsledná vizualizácia pracovala s plnou responzivitou, korektnosťou a plynulosťou. Druhá záložka s názvom Editor predstavuje užívateľské rozhranie tvorby dynamickej prezentácie. Obsahuje v sebe základné interaktívne prvky, ako sú zvýraznenie použitej časti editora, editor aktuálneho elementu a podobne. Tretia záložka s názvom Presentation ukazuje konečnú vizualizáciu tak, ako si ju užívateľ vytvoril v záložke Editor. Prezentácia predstavuje maximálne päť snímok, ktoré sa samostne a dynamicky menia podľa poradia alebo výberu.

V druhej kapitole je prezentovaný prehľad o použití hotových produktov pre bežného užívateľa. V tejto kapitule sú následne spomenuté rôzne knižnice pre prácu s vizualizovaním určitého typu pre webových vývojárov. Je tu spomenutý i výber knižnice pre vývoj bakalárskej práce tak, aby boli podané aj hlavné dôvody výberu.

Tretia kapitola je prezentovaná postupom a výberom elementov, ktoré sa riešili v tejto bakalárskej práci. Nachádza sa tu kompletný manuál pre prácu s nimi. Keďže nie všetky elementy boli podporované pre prácu vizualizácie, je tu oddôvodnenie, prečo element typu video nie je predmetom práce.

Vo štvrtej kapitole je predstavenie aplikácie. Kapitola sa skladá z viacerých častí, ktoré popisujú, ako sa pracuje s jednotlivými elementami a základná, nie úplna, realizácia efektov.

Piata kapitola obsahuje informácie týkajúce sa rozloženia a fungovania editora tak, aby bol užívateľ sám schopný tvoriť dynamické prezentácie. Nachádza sa tu už podrobný popis problémov, realizácie a definovania konkrétnych efektov.

Šiesta kapitola pozostáva z informácií o konečnej prezentácii dynamického charakteru. Prezentuje sa tu charakteristika, obsah a zloženie prezentačného efektu. Je tu vysvetlené vrstvenie a priorita elementov. Rozoberá sa tu možnosť responzivného dizajnu stránky a jeho aplikovanie v aplikácii.

V siedmej kapitole sú realizované testy naprieč modernými internetovými prehliadačmi. Obsahuje informácie o výhodách a nevýhodách jednotlivých prehliadačov a výber odporúčaného prehliadača pre správne fungovanie aplikácie Dynamic Slide Show.

Cieľom bolo docielenie vytvorenia takej dynamickej prezentácie, ktorá bude splňovať všetky požiadavky, aby mala svoje využitie aj v budúcnosti. Dynamický obsah stránok je stále aktuálny,

preto je jasné, že aplikácia sa bude využívať. Napriek nedostatku elementu typu video aplikácia disponuje dostatočným obsahom, aby spĺňala svoju funkcionalitu.

Literatura

[1] B. P. Hogan, *HTML5 a CSS3: Vỳukovỳ kurz webového vỳvojáře.* Computer Press, 2011.

[2] "Powerpoint 2016." `https://products.office.com/sk-sk/powerpoint`. Accessed: 2016.

[3] "Slider revolution." `https://revolution.themepunch.com`. Accessed: 2015.

[4] "About wolframalpha." `https://www.wolframalpha.com/about.html`. Accessed: 2010.

[5] B. Szopka, "impress.js." `https://revolution.themepunch.com`. Accessed: 2011-2016.

[6] H. E. Hattab, "reveal.js." `https://github.com/hakimel/reveal.js`. Accessed: 2016.

[7] M. Bostock, "Data-driven documents." `https://d3js.org`. Accessed: 2015.

[8] "The 38 best tools for data visualization." `https://github.com/impress/impress.js`. Accessed: 11.2.2016.

[9] F. Robichet, "reveal.js." `https://github.com/mbostock/d3/wiki/Selections`. Accessed: 15.4.2016.

[10] M. Bostock, "How selections work." `https://bost.ocks.org/mike/selection/`. Accessed: 26.4.2013.

[11] "Programovanie www stránok - html 5." `http://bc.spsepn.edu.sk/index.php?stranka=programovanie&kap=www&kat=html#html`.

[12] METRIK, "Html." `http://metrik.sk/lexikon/html`. Accessed: 2014.

[13] METRIK, "Css – kaskádové štýly." `http://metrik.sk/lexikon/css-kaskadove-styly`. Accessed: 2014.

[14] METRIK, "Javascript." `http://metrik.sk/lexikon/javascript`. Accessed: 2014.

[15] "jquery - úvod." `http://programujte.com/clanek/2008032701-jquery-uvod/tisk/`.

[16] "Ajax." `http://www.adaptic.cz/znalosti/slovnicek/ajax`.

[17] MSDN, "Asp.net overview." `https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx`. Accessed: 2016.

[18] "Responzívny webdesign." `http://www.psoit.sk/webdesign-responzivny`.